Usage Based Effectiveness Measures

Monitoring Application Performance in Information Retrieval

Leif Azzopardi
Dept. of Comp. Sci., University of Glasgow
Glasgow, United Kingdom
leif@dcs.gla.ac.uk

ABSTRACT

The aim of an Information Retrieval (IR) application is to support the user accessing relevant information effectively and efficiently. It is well known that system performance, in terms of finding relevant information, experienced by the user is heavily dependent upon the application itself (i.e. the IR system exposed through the application's interface) as well as how the application is used by the user (i.e. how the user interacts with the system through the interface). Thus, a very pragmatic evaluation question that arises at the application level is: what is the effectiveness experienced by the user during the usage of the application? To be able to answer this question, we represent the usage of an application by the stream of documents the user encounters while interacting with the application. This representation enables us to monitor and track the performance over time/usage. By taking a stream-based, time-centric view of the IR process, instead of a rank-list, topic/task centric view, the evaluation can be performed on any IR based application. To illustrate the difference and the utility of this approach, we demonstrate how a new suite of usage based effectiveness measures can be applied. This work provides the conceptual foundations for *m*easuring, *m*onitoring and *m*odeling the performance of any IR application which needs to be evaluated over time and in context.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Theory, Experimentation

Keywords

Application, Performance, Usage, User Experience

1. INTRODUCTION

An Information Retrieval (IR) application can comprise of one or more functions/systems exposed through a common interface. For example, a web search engine is typically accessed within a web browser enabling both search, where a ranked list of documents is presented to the user when searching, and browsing of web documents through the hyper-link structure of the web within a single interface. Other examples of information access applications include a news recommender which allows both the push and pull of relevant news; a personal digital libraries that enable indexing, tagging, browsing and retrieval of academic documents; and an advanced intelligent information application that presents a mash up of relevant information from numerous sources in response to a query. A primary goal of all these applications is to enable the efficient and effective access to relevant information. However, when accessing information there are a multitude of ways in which the user can interact with the application. This makes evaluation particularly difficult and a grand challenge for research and development in Interactive Information Retrieval (IIR) [3, 9]. The evaluation difficulties arise because of three main reasons:

1. interaction is difficult to synthesize and replicate making the creation of a TREC like test collection for IIR unfeasible [20];
2. standard IR evaluation [21] focuses on: (a) the system, method or model, and not the application, as well as (b) topic/task, and not the usage over time and the user experience,
3. measurements are typically based on a ranked list with a prescribed order of assessment [19], and this does not generalize well to interactive scenarios [3].

The evaluation of different interactive IR applications, therefore, requires a different evaluation paradigm. One that is more general, at a higher level, and focuses on usage. It needs to be (i) general in the sense that any IR application can be evaluated; (ii) at a higher level, i.e. the application level, to incorporate interaction and the presentation/interface within the evaluation, and; (iii) focused on usage, because how an application is used determines how much relevant information is consumed and thus how effective it is. A paradigm that addresses these needs should enable the evaluation of observational studies of interactive IR to be conducted using the same conceptual framework: in order to *measure*, *monitor*, and *model* the performance experienced during usage.

With recent technological developments (such as Lemur Query Log Project, Google's Search API, Yahoo!'s Search-Monkey Toolkit, etc.), the feasibility of building instrumented interactive IR applications is enabling research to be conducted outside the lab, and in the wild. Consequently, naturalistic and laboratory based observational studies such as [8, 17] are now more viable than ever. This is because as functional IR applications can be cheaply and quickly developed and deployed. This advancement in technology coupled with the move towards evaluating IR systems in context (as exemplified by initiatives like IIiX and the many dedicated workshops on interactive retrieval) means that it is very timely, if not, necessary to now consider a different evaluation paradigm to underpin future IIR research.

To this aim, we adopt a stream based view as advocated by Bookstein [5] to form the basis of evaluation of any IR application: whereby the interaction between the user and the application produces a stream of documents which are assessed during the usage of the application. The sequence of interactions with the application determines the user's perception of the application's performance; which in turn defines their user experience [13]. So from the user's point of view, it is this stream of documents that largely determines their experience of the application [1]. This is because their goal is to access and consume relevant information. Consequently, the application needs to be delivering a sufficient amount of relevant information to the user to have a satisfactory user experience. From an application provider's perspective, it is important that the usage of the application results in a good user experience. As a poor user experience may lead to disengagement or worse abandonment of the application. Given that an application provider wants to retain and expand their user base, then it is important that the usage performance of the application is monitored over time so that such critical incidents can be identified (and minimized). In terms of evaluation at the application level, different questions arise, such as:

1. what is the effectiveness experienced by the user during the usage of the application?
2. how long does it take before the user can effectively operate the application (i.e. burn in or learning effect),
3. how do changes to the application affect the usage performance? and,
4. what level of effectiveness is required to retain users, and at what levels are users likely to disengage or abandon using the application?

Given the motivations for this much needed research, the remainder of this paper will be presented as follows: in the next section, we shall present Bookstein's view of the IR process and use a generalization of the approach to represent any IR application. The stream based view is then used as the basis of measuring the effectiveness of the usage. In Section 3, we formalize the stream based view and define a number of precision based measures along with a novel measure, called Relevance Frequency (the rate at which relevant information is encountered). The main contribution of this work is the formalization of a stream based, time centric view that enables the evaluation of any interactive IR application. Through a series of examples we demonstrate how to employ the proposed measures in a number of different ways in order to measure, monitor and model the performance experienced by the user during the course of interaction with the application.

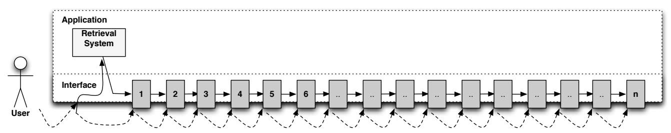


Figure 1: Sequence Diagram of the standard IR Process.

2. A STREAM BASED VIEW

In this section we present the stream based view of the IR process that generalizes to any IR application.

Ranked List / Topic Centric View : First lets review the standard model of the IR process: a user submits a query to the system for a given topic, and the system responds by presenting the user with a ranked list of documents. The user then assesses, in turn, each document in the ranked list. Figure 1 depicts the standard IR process. This ranked list / topic centric view makes several assumptions, regarding the interaction and usage of the application:

- the process is initiated by a query for a given topic (the information need is fixed)
- the documents are presented in a ranked list,
- the user inspects documents, sequentially and in order, and
- the user inspects the entire ranking up to a cut off point n.

This abstraction of the IR process has been the basis of much of the evaluation performed in IR. Consequently, most evaluations perform measurements assuming a ranked list. However, an IR application, and the way it is used, may not necessarily produce a ranked list of documents; nor may the user inspect the documents in a linear fashion; and depending on the application, various functionality may be engaged by the user during the process (such as inspecting document clusters, using a find similar feature, browsing links or facets, etc.). This assumed user behavior is seldom the case, in practice: and so this view of the IR process does not generalize well to non-standard IR applications or non-deterministic usage.

Evaluating interactive IR applications with Ranked lists: One direction that has been taken to evaluate non-standard interaction is to transform the output of the interaction into a ranked list enabling the comparison against standard ranked based methods. For example, in [18], an ostensive browser application recommends similar documents, given the previous documents viewed. The trail of documents the user visits (or the simulated user visits), was used to form a "ranking" which could be compared to a standard retrieval method's ranked lists. Similarly in [11, 12, 16, 23] rankings are formed from the interaction with the envisioned application. While transforming the sequence of documents encountered into a ranked list is appropriate for some applications, it is not possible for all applications. For instance, a filtering application recommends documents, and so the notion that it is evaluated as a ranked list is not appropriate. Nor is it appropriate in an exploratory search application where the query is ill defined, and the information need is highly dynamic. While reverting to a ranked list enables comparison with standard retrieval models, it does not consider evaluation at the application level, where it is important to consider the effectiveness that the user experiences during the usage of the application. We argue that adopting a stream based view provides a general way to represent the usage of an application, such that any IR application can be represented, and that it enables the measurement of the effectiveness experienced throughout the usage of the application.

Stream Based / Time Centric View: The origins of the stream based view stem from the early work of Bookstein [5]. Under Bookstein's view the retrieval process is one where the user examines a sequence of retrieved documents, and where the system receives feedback and adjusts the documents presented to the user. Depending on the feedback, the system presents different documents and the user selections create a particular sequence of retrieved/accessed documents. Bookstein argues that it is this sequence of documents that should be evaluated. In [1], this view is generalized to any IR system, such that the usage of the application results in a stream of documents presented and assessed by the user. For any given IR application, the interaction can be characterized as follows: a user performs an action given the application's interface, the application engages the IR system(s) to produce a response, which is then presented to the user via the application's interface, the user assesses the response and engages with the presented documents, then the user performs a subsequent action, and so on. The order in which the user engages with the presented documents, therefore, defines a stream of documents. This stream based view does not make any assumptions about the user actions/input (i.e. it does not have to be a query), or how the documents are presented to the user (i.e. it does not have to be a ranked list). Without the standard assumptions it is possible to easily generalize the stream based view to consider any type of IR application. Some example IR applications are shown in Figure 2 which are all represented using this view, where the interaction in the sequence diagram shows the stream of documents (denoted by $1, \ldots, n$) built up over the course of interaction with the application.

The main difference between this view and the standard rank based view, is that it is temporal and usage specific: where the stream of documents encountered by the user depends upon on how the application was used. It is this stream that forms the basis of the evaluation of the effectiveness that the user experiences during the usage of the application. While the stream based view imposes little restrictions on describing the IR process, it does require other assumptions to be engaged:

1. only one document can be accessed/viewed at a time, and
2. each document is independently judged, each time, that it is accessed.

The first assumption is that the user can only access one document at any particular point in time. There are instances when parallel streams of documents are encountered during the usage of the IR application (this may be the case when dealing with images, or comparing multiple documents in multiple windows). Nonetheless, it is a reasonable assumption that the user will only be examining one particular document at time.

In the standard view, it is assumed that each document is judged (independently) with respect to a fixed topic. However, it has been widely acknowledged that through interaction, the information need changes as the user's state of knowledge changes. Under the stream based view, it is assumed that each document is judged with respect to the user's current information need/state. Each judgement is based on the current information need, and given the current (*perhaps altered*) state of the user. This is a very important point, because it means that when the same document appears in the stream, it may attract a different relevance judgement by the user.

These assumptions mean that it is the usage of the application that is evaluated and not whether the system is able to achieve total recall. Thus, usage based effectiveness measures will be predominately precision oriented.

Summary Taking a stream based view allows the usage based effectiveness of any IR application to be evaluated using the same conceptual basis. The same question can be asked of any IR application, such as web browsers, news recommendation applications, search engines, query-less ostensive browsers, etc, that is: how effective is the usage of the IR application? While the standard rank-based view enables a thorough comparison of the different systems/models to be achieved under prescribed circumstances, the stream based view enables the measuring and monitoring of the performance of the usage of the application. Invariably, the usage performance is going to be a result of the (potentially synergistic) interaction between the user and the application. This view means that either a observational or simulated study needs to be undertaken in order to evaluate the usage performance experienced over time by the user.

3. USAGE PERFORMANCE

In this section, we shall describe a suite of measures for tracking and monitoring the performance resulting from the usage of an application under the stream based view. These measures are designed specifically to be used in a simulation or observational study: so that the effectiveness of the usage of various interactive IR applications can be analyzed and modeled.

While there are many possible measurements that can be exploited, here we focus on using precision oriented measurements taken on sub-streams of documents in order to form an estimate of the application performance at a particular point in the stream, or particular time period within the stream. Since an application's performance is monitored in the context of usage, it is not possible to develop recall based measures. This would require a post-hoc assessment of all documents. Such a task is problematic due to changes (or the evolution) of a user's information need over time and context [7]. We then introduce a novel measure of the performance, called *relevance frequency*: which characterizes the rate at which relevant information is encountered during the usage of an IR application. First, we begin with the necessary definitions and notation, before outlining these different measures.

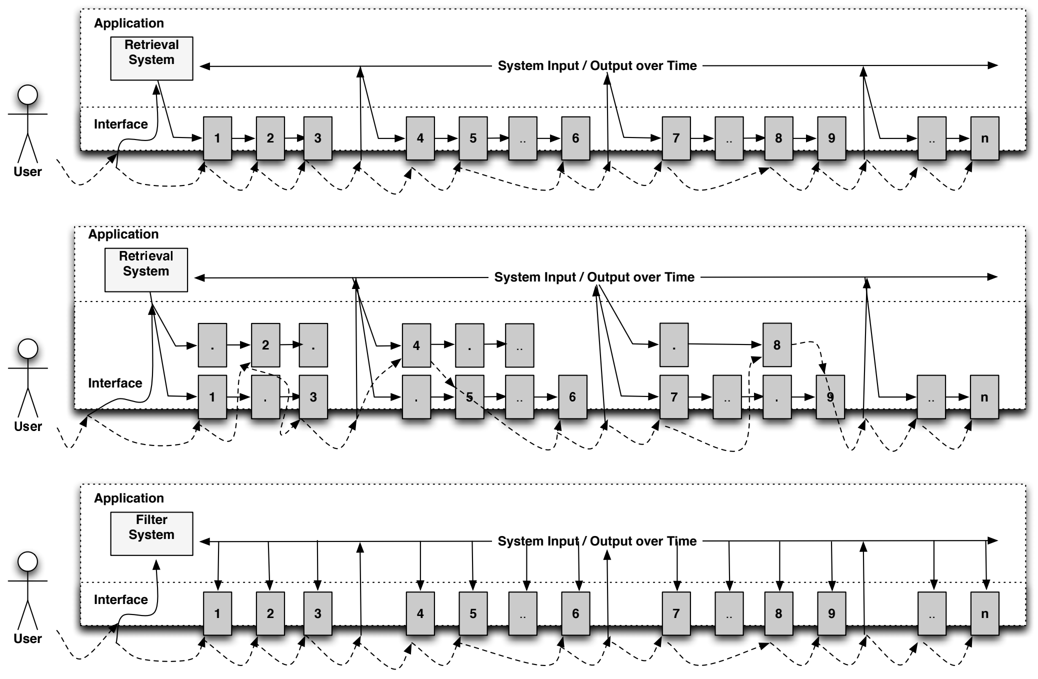


Figure 2: Sequence Diagrams for an Interactive Retrieval, Cluster Based Retrieval, and Filtering applications (top, middle, and bottom). The dotted line denotes the user interacting with the application. The result is the stream of documents which are assessed during the usage of the application.

3.1 Preliminaries: Notation and Definitions

Before we outline the notation, it is necessary to clarify the definitions of streams and sub-streams. A stream is a sequence of objects ordered temporally. A sub-stream is a sub-sequence derived from the stream, where the order of the objects is preserved. For the purposes of measurement the stream is decomposed into sub-streams. This can be performed, logically, conceptually or practically depending on the specific unit of interest (i.e. topic,session, hour, day,etc).

To formalize the measurement of streams we first introduce some notation. Let $\mathbf{s}$ denote a (sub) stream which consists of a sequence of documents such $\mathbf{s} = (d_1, d_2, \ldots, d_N)$ with length N. For each document d_i we assume that there is an associated judgement r_i assigned to it forming a corresponding sequence $\mathbf{r} = (r_1, r_2, \ldots, r_N)$. A stream s can be decomposed into sub streams, such that $s_{ij} = (d_i, \ldots, d_j)$, we shall use stream and sub-stream interchangeably.

For the purpose of introducing the set of measures we focus on a dichotomous decision based on relevance (e.g. document relevance or utility)[1], where $r_i = \{0, 1\}$. However, the value of value r_i could also be a rating, grade or a continuous measurement. This judgement represents whether the document is relevant or useful to the user at that time point in the stream.

For a given stream $\mathbf{s}$ with the corresponding sequence of judgements $\mathbf{r}$, it is possible to estimate the precision of the stream by treating the stream as a set and determining the proportion of relevant documents within the stream:

$$\text{Prec}(\mathbf{s}) = \frac{1}{N} \sum_{i=1}^{N} r_i \tag{1}$$

where

$$r_i = \begin{cases} 1, & \text{if } d_i \text{ is relevant} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Given a series of sub-streams ordered by time, it is then possible to obtain a series of precision measurements across the entire stream. While the individual sub-streams are treated like sets, the order of the measurements determines the usage performance experienced over the course of interaction.

3.2 Precision Stream Measures

Depending on the type of application and the focus of the evaluation the stream is decomposed into sub-streams, accordingly. This defines the unit of measurement. While there are many possible ways to decompose the stream, here we only consider a few possible variations (and leave other variations for future work):

- **Precision Blocks:** the stream is decomposed as contiguous sub-streams $\mathbf{s_{ij}}$ of equal length N.
- **Precision Windows:** A stream is decomposed into overlapping sequences of equal size N. In other words a window is moved across the stream to create sub-streams (i.e. $\mathbf{s}_{ij}, \mathbf{s}_{i+1,j+1}, \mathbf{s}_{i+2,j+2}, \ldots$).
- **Precision Day/Week/Month:** A stream is decomposed into contiguous sub-streams according to a time

[1]This does not restrict the decision to relevance but could be any question posed to the user.

unit such as hour, day, week, month, etc, resulting in sub-streams of different length (as the number of documents accessed in a given time frame may vary).

- **Precision Session/Topic:** A stream is divided into sequences of variable length determined by a (user) session or topic. i.e. streams can be decomposed such that they are ranked lists, where precision stream measures can be applied, as well as numerous other ranked lists based evaluation measures (if desired).

Regardless of the decomposition, these measures will provide an indication of the application's performance over time, enabling the monitoring of system performance. The main distinction between the Block and Window measures and the other measures is that they do not consider the time between document interactions, but simply the order, while the other measures consider the period of time in which the usage took place. As we shall see (in the next section) both provide interesting ways in which to track, monitor and analyze the usage performance.

Cumulative Average Precision: As the precision measures provide point estimates of performance for the sub-streams, it is of interest to summarize the usage performance experienced over these sub-streams (i.e. a temporal average). The *cumulative (marco) averaged precision* (CAP) can be obtained by averaging over the measurements taken on a stream $\mathbf{s}$ decomposed into M sub-streams $\mathbf{s}_j$, as follows:

$$CAP(S \leq \mathbf{s_M}) = \frac{\sum_{j=1}^{M} Prec(\mathbf{s}_j)}{M} \tag{3}$$

this represents the cumulative distribution of precision in the stream up to and including sub-stream $\mathbf{s}_M$. We refer to this as a macro-average since the average is taken based on the precision of the sub-streams, and differs from the micro-average which be estimated at the document level. Measurements of this nature indicate how the application's usage performance convergences over time/usage. Other statistics, such as the standard deviation and standard error for the stream precision measurements can also be calculated.

3.3 Relevance Frequency

In the previous subsection, we defined a suite of precision based measures, however, this only provides one possible way to evaluate streams. In this subsection, we propose a novel stream based measure which conveys a different view of the usage performance. Intuitively, the number of documents a user must examine before encountering a relevant document will impact upon their user experience. If they encounter many non-relevant documents successively this will detract from the user experience. Many long periods of non-relevance before encountering relevant information is likely to lead to a negative user experience. The Relevance Frequency measure aims to quantify the rate at which relevant documents are encountered during the stream.

RFreq: Given a stream $\mathbf{s}$, which is decomposed into sub-streams, by slicing the stream up, whenever a relevant document occurs. The length x of a sub-stream denotes how many documents are examined to find a relevant document. So the Relevance Frequency for a distance of x is the count of the number of sub-streams that are of length x, ($RFreq(x)$). For example, if the judgements on a stream yields:

$$\{|R|R|N, R|N, N, R|N, N, N, R|\},$$

where the $|.|$ indicates the sub-streams. Then, the $RFreq(1) = 2$ because there are two sub-streams of length one, and so on, such that: $RFreq(2) = 1$, $RFreq(3) = 1$, $RFreq(4) = 1$, and $RFreq(> 5) = 0$. By plotting the $RFreq(x)$ values the distribution of encountering relevant documents can be visualized (see Figures 3 and 6 for examples of RFreq plots.)

Points of Failure (*pof*): If it is crucial that the application delivers relevant information at least every y documents, then it is easy to compute the number of points in the stream where this criteria is not satisfied, i.e. the number of points of failure (pof) is equal to the:

$$pof(x > y) = \sum_{x>y} RFreq(x) \tag{4}$$

Table 3 provides an example, where the $pof(x > 10) = 163$. This means that during the usage of the application there are 163 critical instances where the user has to endure at least ten documents before encountering a relevant document. Whether these periods of prolonged non-relevance are tolerated depends upon the user and the application. Obviously, a stream $\mathbf{s}$ which contains all relevant documents will results in a $RFreq(x = 1) = n$, where n is the length of the stream, and $RFreq(x > 1) = 0$. Whereas a stream which contains only non-relevant documents will result in an $RFreq(x = i) = 0$ for all $i \leq n$. This is because no relevant document occurs in the stream, so no sub-streams which are terminated by a relevant document exist within $\mathbf{s}$.

EFreq: To summarize the distribution of the Relevance Frequencies, the maximum likelihood estimate of the distribution can be taken to obtain the Expected Relevance Frequency as follows:

$$E[RFreq] = \frac{\sum_{x} x \times RFreq(x)}{\sum_{x'} RFreq(x')} \tag{5}$$

where $E[RFreq]$ denotes the expected rate of relevant documents in the stream. If $E[RFreq] = 10$ on average, the user could expect to encounter nine non-relevant documents before encountering a relevant document at the tenth position, on average. An interesting direction for future work would be to model the Relevance Frequency as a probability distribution in order to summarize and compare usage performance.

4. EMPIRICAL DEMONSTRATION

To demonstrate the utility of the usage based effectiveness measures we shall illustrate the application of such measures using a number of different scenarios. During the course of this demonstration we shall show examples of:

(i) how the stream based view can be used to monitor the performance of different IR applications (see §4.2),

(ii) how the measures can be used to describe the performance experienced by the user when interacting with different IR applications, specifically:

  - how different user behaviors impact upon the usage performance (see §4.3)
  - how different systems impact upon the usage performance (see §4.4), and,

– how the usage can be modeled with the relevance frequency measure (see §4.5)

4.1 Experimental Set-up

For the purposes of this conceptual paper, we shall simulate a number of different IR scenarios that use different IR applications in order to demonstrate the proposed measures.

Methods and Materials: The empirical setup of the demonstration uses a TREC test collection for which we have a number of topics and corresponding relevance judgments. While these topics and judgments have been created in a specific manner, given the more liberal modeling assumptions of the stream based view, this test collection will enable us to demonstrate the application of usage based effectiveness measures *in a simulated setting*. It should be noted that simulation provides a powerful tool for hypothesizing about user behavior [12, 23]. The analysis is conducted using the AP 88-89 TREC test collection with topics 51-100. The collection was indexed in Lemur[2], where stop words were removed and stemming applied. The applications considered are as follows:

- a retrieval application with a fixed mode of interaction (representing the traditional IR process, but under the stream based view),
- a retrieval application with variable interaction (representing an interactive retrieval scenario, where we simulate a user berry picking during their usage of the retrieval application [2]), and,
- a filtering application, which recommends news documents to the user over time.

While filtering and retrieval are considered to be two sides of the same coin they are evaluated differently[4]. Retrieval uses a ranked list as the basis of measurement, while in filtering the set of filtered documents forms the basis of measurement (as done in the TREC Filtering Tracks, e.g. [14]). Under a stream based view, conceptually they are measured in the same way though the interpretation of the results is different. It should be noted that the usage of the filtering application occurs over many weeks, months, years, whereas the usage of the retrieval application could occur over a much shorter time frame. Thus for this demonstration, we can only show the performance of the retrieval applications over blocks and windows, where as for the filtering application we can show the performance over both blocks/windows and day/week/month/etc. For brevity and clarity, we shall only report the following measures: Block Precision (BP) and Week Precision (WP), along with Relevance Frequency (RFreq), and their associated averages.

Standard Retrieval Application: Each query was posed to the retrieval application, and the top 200 documents retrieved were concatenated together to form a stream based on the 50 TREC topics. Essentially, we assume the user proceeds to perform a search on each topic, sequentially. We further assumed that the user examines the top 200 documents per topic (i.e. a fixed form of interaction), and that each topic is queried in numeric order. Given this scenario, it is interesting to note the difference between BP and Precision at n documents, as they are very similar. Table 1 shows performance on a stream defined by a ranked list for

[2]`http://www.lemurproject.org`

n	block	rels	**P@n**	**BP**	**CAP**
25	1	15 (15)	0.6	0.6	0.6
50	2	10 (25)	0.5	0.4	0.5
75	3	5 (30)	0.4	0.2	0.4
100	4	0 (30)	0.3	0.0	0.3
125	5	5 (35)	0.28	0.2	0.28

Table 1: BP, P@n and CAP for a query stream aka ranked list. The BP reflects the current effectiveness experiences when they examine each page of results, where as the $P@n$ reflects the average effectiveness over n. Only first five pages of results shown.

P@n and BP at intervals/blocks of 25 documents. The BP reflects the current effectiveness the user experiences when they examine a given page of 25 results, where as the $P@n$ reflects the average effectiveness over n. It is of note that the P@n and CAP provide the same value, *in this specific case.* This is because the blocks are of equal size, so the macro averaging over blocks is the same as the micro average over n documents, and because we are evaluating the ranking in a stream based view. For Day/Week/Month/etc Precision measures where the length of the sub-streams is variable then the micro and macro averages will be different. The added advantage of BP (and the other stream based precision measures) is that the variability of the performance, such as the standard deviation, can also be obtained because the average is taken across a number of blocks (e.g. see Figure 5 for an example using WP).

Retrieval Application with Interaction: To simulate a different usage of the retrieval application, we shall assume that the user takes a "berry picking" like approach [2], where they only harvest the tops of ranked lists for relevant documents (instead of trawling through every single document in the ranking). We shall further assume that after querying the user is presented with a page of 25 results (i.e. a block size = 25), the user assesses these documents. If the performance of the current page is above a user defined threshold (λ) then they continue to the next page of results, otherwise they curtail the search and proceed to issue the next query.

Filter Application: For the filtering application, we assume that a user sets up a number of filters within the news recommender filtering application. As news stories are received by the application, if they appear relevant to one of the filters then it is recommended to the user. The stories are presented to the user on a daily basis as the news breaks. In our example, we monitor the weekly performance and for the purposes of illustration, we used the same 200 documents retrieved for each topic in retrieval application scenario. However, we ordered the documents by their publication date to simulate the stories being recommended when the news breaks. This was done to show that different orderings of the same set of documents result in distinctly different usage performance experienced over time.

4.2 Example One: Monitoring Performance

How does the performance of different IR applications vary during their usage? For retrieval, previous research suggests that as the user evaluates more of the ranked list, the precision will decrease (as seen when examining the precision at k documents measurements). However, for other

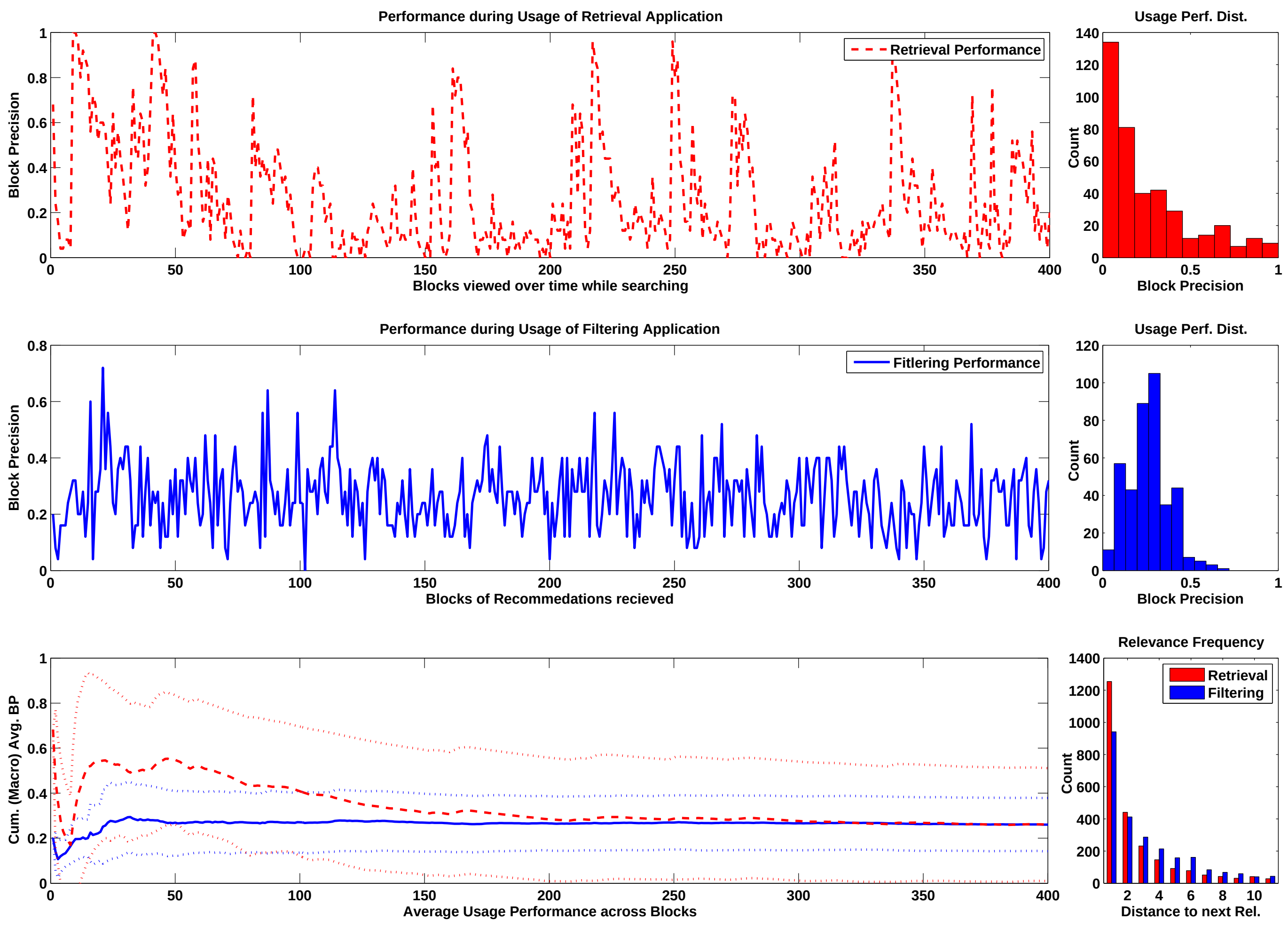


Figure 3: The performance for retrieval (top plot) and filtering (middle plot) applications for BP (left) and Dist. of Usage Performance (right). The bottom left plot shows the cum. Avg. BP, along with std. dev. (dotted lines), and the bottom right plot shows the Relevance Frequency for both applications.

applications, like filtering, it is not known. For this demonstration, we show the usage performance for the retrieval and a filtering application to illustrate the differences in the effectiveness experienced by the user.

Figure 3 shows the BP measured on blocks of 25 documents within the streams of the retrieval application (top plot) and the filtering application (middle plot). While the precision given the entire stream produced by both applications is the same in this constructed example, the precision experienced at different blocks in the stream is considerably different. Usage of the standard retrieval application in this way, results in the user experiencing very high precision usually after they enter a query, followed by low precision as they examine the ranking provided. On the other hand, the usage of the filtering application results in the user experiencing less variation in usage performance. The two additional plots on the right show the distribution of the Block Precision over the observed period. The retrieval application's usage results in performance following an exponential distribution, while the filtering application's usage results in performance following a relatively normal distribution.

The bottom-left subplot, in Figure 3 shows the cumulative averaged BP up to the ith block (along with standard deviation denoted by the dotted lines). This figure clearly shows that the performance of a filtering application converges to the average performance much quicker than the performance of the retrieval application. This is due to the performance for filtering being sampled from a "normal" distribution. From these plots, it is quite clear that the performance experienced over time varies depending on the application.

The bottom-right subplot shows the Relevance Frequency distribution for each application. The expected Relevance Frequency is 3.279 and 3.806 for the retrieval and filtering applications, respectively. Table 3 also provides statistics to show that while the retrieval application has a lower expected RFreq, there are more times when the user will have to endure 20 or more documents, than when using the filtering application (34 versus 16). This is because the retrieval application delivers more relevant content at the beginning of the ranked list, and less relevant content at the end of the ranked list.

4.3 Example 2: Interaction on Performance

Another type of analysis that can be conducted using stream based measures is to examine how the application performance changes due to the user behavior. Here, we provide a very simple demonstration, where we assume that a simulated user adopts a "berry picking" information seeking behavior [2]. Given the retrieval application (as described above), we assume that a user poses a query, and the system responds with a page of results (a block of 25 documents) which the user examines. If the page is sufficiently rich in relevant material, this motivates the user to continue their search and examine the following page. Otherwise, the user will stop examining the current result list and pose the next query given the set of topics. While we have artificially constructed this example, it captures the essence of the berry picking approach, as the user only continues the results are rich in relevant information.

For this experiment, we varied the threshold λ between 0 and 1. The threshold indicates the decision criteria for our simulated user to continue to the next page (i.e. block), or not. If the current page's precision is higher than the threshold λ, the user continues, else they stop and move on to the next query in the topic set. A threshold of zero means the user examines all pages, and one indicates that the user will examine only the first page of results for each query (which is equivalent to Precision at $k = 25$). Table 2 shows the performance experienced over the course of interaction with the retrieval application for different thresholds, along with reporting the number of blocks/pages of results viewed during the usage, and the Average number of relevant documents per block.

When the user examines all pages ($\lambda = 0$), then overall usage performance experienced is 0.26 BP. Whereas if the user examines only the first page ($\lambda = 1.0$) then the overall usage experienced is 0.45 BP. However, the best usage performance possible is obtained when the user's decision criteria is in between these two extremes – and at $\lambda = 0.7$ the BP is 0.55! Thus, a savvy user can work the application in order to maximize the usage performance (and as shown in [15], users can overcome system side deficiencies by compensating through different interactions and strategies). To further illustrate the point, Figure 4 shows the distribution of Block Precision for three cases ($\lambda = \{0, 0.2, 0.5\}$). The results appear to indicate that the berry picker strategy reduces the number of times the user encounters pages with only few relevant documents: resulting in the user experiencing higher effectiveness during the course of usage. Since the number of documents accessed is less because of their interaction, it is a open question whether a user could maintain or sustain this level of effectiveness if they continued querying.

4.4 Example 3: System on Performance

In this example we shall use the filtering application example where we monitor the weekly usage performance. Figure 6 shows the Weekly usage performance for the filtering application powered by either BM25 or LM. In this simulated environment it is possible to obtain the performance for both algorithms; however, if an application is deployed in the wild, it is not possible to perform such a paired comparison. Instead, a classical observational experiment needs to be conducted: observe without the experimental condition, observe with experimental condition, and observe without the experimental condition (i.e. O1 X O2) [6].

For the first 30 weeks, the application is configured using BM25 (BM), then for the next 30 weeks, a poorly tuned Language Model (LM) is used, then for the remaining 30 weeks of usage, BM25 is used. The weekly precision is obtained for each week, and the macro-averaged Week Precision is also computed for weeks 1-30,31-60 and 61-90, along with the standard deviation. Figure 5 shows the plots of the Week Precision and the cumulative average weekly precision. It is quite clear that the mean performance of the BM powered application outperforms the LM one. By performing an unpaired statistical test it is possible to determine whether there is any significant between the underlying methods.

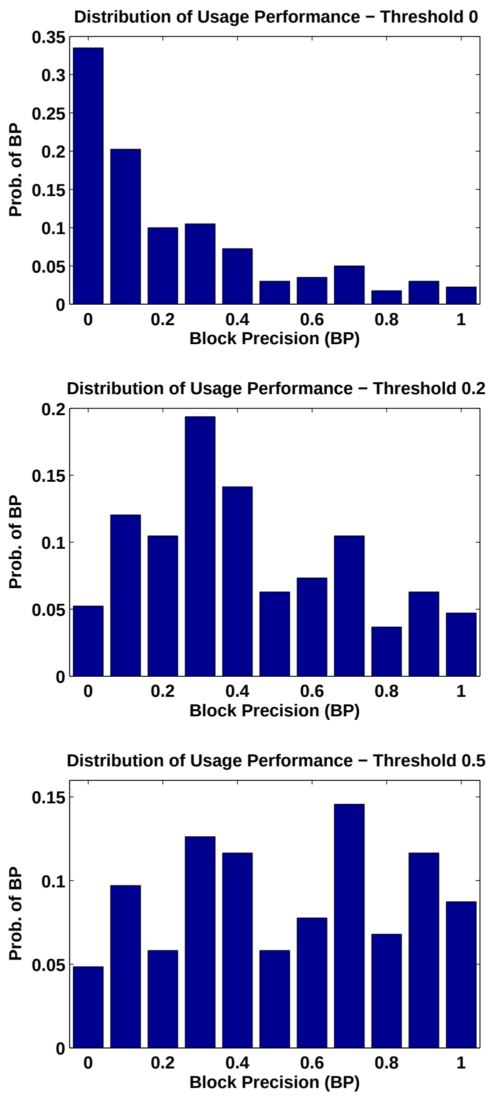


Figure 4: Distribution of Usage Performance (Block Precision) for the different berry pickers.

4.5 Example 4: Modeling Performance

To demonstrate the Relevance Frequency measure, we again use the filtering application. In this example, we use four filters, BM and LM, as above, and a TFIDF filter (TF), and also BM25 with pseudo relevance feedback (BMFB) filter. The IR application is configured with each of the different filters, and then monitored across the 90 weeks of the AP88-89 collection given the 51-100 topic set.

Figure 6 shows a plot of the Weekly Precision for BM and LM filtering applications, along with the Relevance Frequency plot. In Table 3 the expected Relevance Frequency,

Threshold λ	Avg. BP	Num. Blocks Viewed	Avg. Num. Rels. / Block.
0	0.26	400	6.51
0.1	0.364	237	9.10
0.2	0.440	191	10.99
0.3	0.475	155	11.88
0.4	0.526	113	13.15
0.5	0.533	103	13.32
0.6	0.542	91	13.54
0.7	**0.555**	**83**	**13.88**
0.8	0.539	66	13.47
0.9	0.512	58	12.81
1.0	0.450	50	11.24

Table 2: Summary Performance statistics of different berry pickers given the same retrieval application.

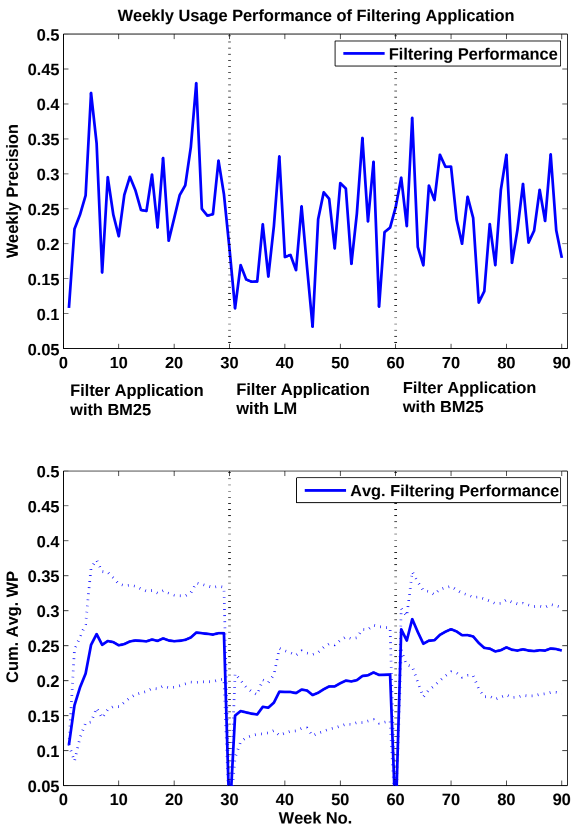


Figure 5: An Observational Experiment: where the application uses a BM filter (Weeks 1-30), an LM filter (Weeks 31-60), and then the BM filter (Weeks 61-90). The top plot shown the Week Precision experience over time, and the bottom plot shows the CAP and standard deviation (dotted lines) for each 30 week period.

and the number of times the application fails to deliver a relevant document after seeing 10 or more and 20 or more non-relevant documents, respectively is also shown. The Expected Relevance Frequency indicates that BMFB method

	Application				
	Ret.	Filter Config.			
Measure	BM	BM	BMFB	TF	LM
$E[RFreq]$	3.38	3.81	3.64	4.03	4.38
$pof(x > 10)$	163	178	147	184	204
$pof(x > 20)$	34	16	9	23	36

Table 3: Relevance Frequency Summary statistics of the different IR applications' usage.

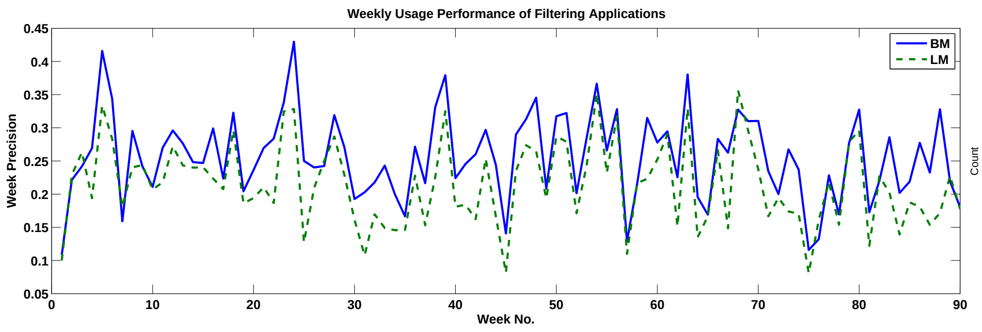
delivers, on average, one relevant documents per 3.64 document, where as the LM method delivers, on average, one relevant document per 4.38. We can also see that there are 36 times when the LM method fails to deliver a relevant document after encountering 20 or more non-relevant documents. On the other hand, for BMFB this drops to only 9 such incidents. From an application provider's point of view, the higher the RFreq and the lower the number of times the user has to experience prolonged periods of non-relevance, the better: and this is likely to translate into a better user experience. However, it is a matter of empirical investigation to determine whether such measures correlate/correspond to a good user experience, or not. Though intuitively this would appear to be the case.

5. DISCUSSION AND SUMMARY

In this paper, we have formalized an alternative paradigm for the evaluation of IR applications: this stream-based / time-centric view represents the usage of any IR application through a stream of documents. This stream enables the usage performance to be evaluated. Given the goal of an IR application, the proposed usage based effectiveness measures provide a novel way in which to monitor and model the performance experienced by the user while interacting with the application. However, there are two main practical considerations that need to be addressed in order to monitor the usage performance of an application:

- how to build up the stream of documents (and how to deal with un-assessed but presented documents, etc) i.e. what documents go into the stream, and:
- obtaining judgements on the utility/relevance of each document encountered in the stream, implicitly and unobtrusively.

For the purposes of introducing stream based measures, we have assumed that every document in the stream has a corresponding judgement (i.e. the completeness assumption), however this does not mean that the stream can not contain un-assessed documents. But considering un-assessed documents would require further measures to be developed. These could reflect other aspects of the user experience, such as user engagement. The second consideration poses the greater challenge. However, there have been many advancements made towards developing mechanism to infer the relevance of documents implicitly [10, 8, 22]. And, as these mechanisms improve and the implicit judgements become more reliable, the quality and accuracy of the usage performance measures will also improve making this form of evaluation feasible.



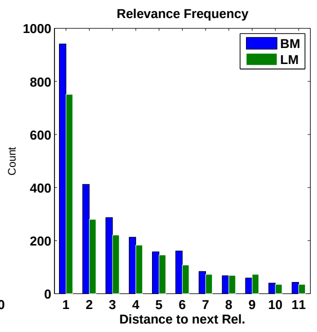


Figure 6: Comparison via Simulation: Weekly Performance (left plot) and the Relevance Frequency (right plot) for the filter application configured with BM and LM methods.

In this paper, we have only been able to explore some aspects of the stream-based view, however, there are many aspects that we have not discussed or explored here, but which motivate further research, such as:

- the development of a suite of stream specific measures like Relevance Frequency, and averaging over streams, for instance by drawing upon signal processing techniques such as the Hamming window,
- how to compare streams of different lengths, and the extrapolation of performance over time,
- how to compare different systems/users, and how to model and predict usage performance.
- how well do the proposed measures correlate to the user experience? and what is the level of performance required before a user will/would adopt an application, and at what critical point would the user dis-engage with or abandon an application?

These questions are left for future work.

In conclusion, the stream based view naturally focuses evaluation on the usage of an application, and thus requires observational experiments: either simulation based or with users, in the lab or in the wild. With recent developments in technology to rapidly build and prototype practical and real world IR applications then the number of naturalistic studies is likely to increase. So, engaging a stream based view in order to measure, monitor and model the usage performance of IR applications is not only timely, but necessary to facilitate the evaluation of these future interactive IR applications. Finally, the stream based view entails a shift away from evaluation through collections (especially as this is considered impractical [20]), and towards evaluation through applications deployed in the wild (i.e. Science 2.0).

Acknowledgements: I would like to thank Mark Baillie for his help in the initial stages of this work and suggestions on modeling performance (which we did not manage to fully develop here).

6. REFERENCES

[1] L. Azzopardi. Towards evaluating the user experience of interactive information access systems. In *Proc. of WISI Workshop at SIGIR 2007*, 2007.
[2] M. J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424, 1989.
[3] N. J. Belkin. Some(what) grand challenges for information retrieval. *SIGIR Forum*, 42(1):47–54, 2008.
[4] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.
[5] A. Bookstein. Information retrieval: A sequential learning process. *JASIS*, 34(5):331–342, 1983.
[6] D. T. Campbell and J. C. Stanley. *Experimental and Quasi-Experimental Designs for Research*. Rand McNally & Co., 1966.
[7] S. P. Harter. Psychological relevance and information science. *JASIS*, 43(9):602–615, 1992.
[8] D. Kelly. *Understanding implicit feedback and document preference: a naturalistic user study*. PhD thesis, Rutgers University, New Brunswick, NJ, USA, 2004.
[9] D. Kelly, S. Dumais, and J. O. Pedersen. Evaluation challenges and directions for information-seeking support systems. *Computer*, 42(3):60–66, 2009.
[10] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.
[11] A. Leuski. Evaluating document clustering for interactive information retrieval. In *Proc. of CIKM '01*, pages 33–40, 2001.
[12] J. Lin and M. D. Smucker. How do users find things with pubmed?: towards automatic utility evaluation with user simulations. In *Proc. of SIGIR '08*, pages 19–26, 2008.
[13] D. A. Norman. *The Design of Everyday Things*. Doubleday, New York, 1988.
[14] S. Robertson and I. Soboroff. The TREC-10 filtering track report. In *Proc. of TREC-10*, 2001.
[15] C. L. Smith and P. B. Kantor. User adaptation: good results from poor systems. In *Proc. of SIGIR '08*, pages 147–154, 2008.
[16] M. D. Smucker and J. Allan. Using similarity links as shortcuts to relevant web pages. In *Proc. of SIGIR '07*, pages 863–864, 2007.
[17] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proc. of CHI '04*, pages 415–422, 2004.
[18] J. Urban, J. M. Jose, and C. J. van Rijsbergen. An adaptive approach towards content-based image retrieval. In *Proc. of CBMI'03*, pages 119–126, 2003.
[19] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.
[20] E. M. Voorhees. On test collections for adaptive information retrieval. *IPM*, 44(6):1879–1885, 2008.
[21] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.
[22] R. W. White and D. Kelly. A study on the effects of personalization and task information on implicit feedback

performance. In *Proc. of CIKM '06*, pages 297–306, 2006.

[23] R. W. White, I. Ruthven, J. M. Jose, and C. J. van Rijsbergen. Evaluating implicit feedback models using searcher simulations. *ACM TOIS*, 23(3):325–361, 2005.